

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日 2 0 0 3 年 1 月 9 日
Date of Application:

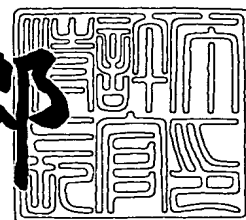
出 願 番 号 特 願 2 0 0 3 - 0 0 3 4 2 8
Application Number:
[ST. 10/C]: [J P 2 0 0 3 - 0 0 3 4 2 8]

出 願 人 株式会社東芝
Applicant(s):

2 0 0 3 年 7 月 8 日

特許庁長官
Commissioner,
Japan Patent Office

太田信一郎



出証番号 出証特 2 0 0 3 - 3 0 5 4 0 1 2

【書類名】 特許願

【整理番号】 13890301

【提出日】 平成15年 1月 9日

【あて先】 特許庁長官殿

【国際特許分類】 H06L 9/10

【発明の名称】 プロセッサ、実行タスク決定装置及び演算処理方法

【請求項の数】 17

【発明者】

 【住所又は居所】 神奈川県川崎市幸区小向東芝町 1 番地 株式会社東芝
研究開発センター内

 【氏名】 鎧 木 智

【発明者】

 【住所又は居所】 神奈川県川崎市幸区小向東芝町 1 番地 株式会社東芝
研究開発センター内

 【氏名】 宮 本 幸 昌

【発明者】

 【住所又は居所】 神奈川県川崎市幸区小向東芝町 1 番地 株式会社東芝
研究開発センター内

 【氏名】 菅 野 伸 一

【発明者】

 【住所又は居所】 神奈川県川崎市幸区小向東芝町 1 番地 株式会社東芝
研究開発センター内

 【氏名】 樽 家 昌 也

【発明者】

 【住所又は居所】 神奈川県川崎市幸区小向東芝町 1 番地 株式会社東芝
研究開発センター内

 【氏名】 大根田 拓

【特許出願人】

【識別番号】 000003078

【住所又は居所】 東京都港区芝浦一丁目1番1号

【氏名又は名称】 株式会社 東 芝

【代理人】

【識別番号】 100075812

【弁理士】

【氏名又は名称】 吉 武 賢 次

【選任した代理人】

【識別番号】 100088889

【弁理士】

【氏名又は名称】 橋 谷 英 俊

【選任した代理人】

【識別番号】 100082991

【弁理士】

【氏名又は名称】 佐 藤 泰 和

【選任した代理人】

【識別番号】 100096921

【弁理士】

【氏名又は名称】 吉 元 弘

【選任した代理人】

【識別番号】 100103263

【弁理士】

【氏名又は名称】 川 崎 康

【手数料の表示】

【予納台帳番号】 087654

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 プロセッサ、実行タスク決定装置及び演算処理方法

【特許請求の範囲】

【請求項 1】

複数のタスクを時分割で処理するプロセッサにおいて、
予め定めたタスクに対して、該タスクの処理で使用されるデータを記憶する 1 以上の第 1 の FIFO (First In First Out) 型記憶手段と、
前記第 1 の FIFO 型記憶手段から取得したデータを使用して前記タスクの処理を行った後の処理結果を記憶するとともに、この記憶した処理結果を使用する他のタスクがある場合には該他のタスクの処理で使用されるデータを記憶する 1 以上の第 2 の FIFO 型記憶手段と、
前記第 1 の FIFO 型記憶手段が前記タスクの処理に使用するデータを保持しているか否か、および前記タスクの処理結果を記憶する空き領域を前記第 2 の FIFO 型記憶手段が有しているか否かを判断するタスク判断手段と、
前記タスク判断手段の判断結果に基づいて、前記複数のタスクのうちから次に起動すべきタスクを決定する実行タスク決定手段と、を備えることを特徴とするプロセッサ。

【請求項 2】

前記複数のタスクのそれぞれに優先順位を設定する優先順位設定手段を備え、
前記実行タスク決定手段は、前記優先順位設定手段により設定された優先順位に基づいて、前記起動すべきタスクを決定することを特徴とする請求項 1 に記載のプロセッサ。

【請求項 3】

前記複数のタスクそれぞれの起動頻度を計測する起動頻度計測手段を備え、
前記優先順位設定手段は、前記起動頻度計測手段で計測された起動頻度に基づいて、前記優先順位を設定することを特徴とする請求項 2 に記載のプロセッサ。

【請求項 4】

前記優先順位設定手段は、前記起動頻度がより高いタスクほど優先順位を高く設定することを特徴とする請求項 3 に記載のプロセッサ。

【請求項 5】

前記実行タスク決定手段は、第 1 のタスクの起動中に、該第 1 のタスクよりも優先順位の高い第 2 のタスクに使用されるデータを前記第 1 のFIFO型記憶手段が保持し、かつ前記第 2 のタスクの処理結果を格納すべき空き領域を前記第 2 のFIFO型記憶手段が有する場合、前記第 1 のタスクの起動を中断して、前記第 2 のタスクを起動することを特徴とする請求項 2 に記載のプロセッサ。

【請求項 6】

個々のタスクに固有の情報をすべて含む 1 以上のレジスタからなるレジスタセットを、時分割で実行可能なタスクの総数以上設けたレジスタセット群と、

前記実行タスク決定手段により決定したタスクが利用する前記レジスタセットを選択するレジスタセット選択手段と、

前記レジスタセット選択手段で選択したレジスタセットを利用して、前記実行タスク決定手段により決定したタスクの処理を行う演算処理手段と、を備えることを特徴とする請求項 1 に記載のプロセッサ。

【請求項 7】

タスクと、該タスクが利用する前記レジスタセットと、の対応関係を記憶する対応関係記憶手段を備え、

前記レジスタセット選択手段は、前記対応関係記憶手段に記憶した対応関係に基づいて、前記実行タスク決定手段によって決定したタスクが利用するレジスタセットを選択することを特徴とする請求項 6 に記載のプロセッサ。

【請求項 8】

個々のタスクに固有の情報をすべて含む 1 以上のレジスタからなるレジスタセットを、時分割で実行可能なタスクの総数に満たない数だけ設けたレジスタセット群と、

前記実行タスク決定手段により決定したタスクが利用する前記レジスタセットを選択するレジスタセット選択手段と、

前記レジスタセット選択手段で選択したレジスタセットを利用して、前記実行タスク決定手段により決定したタスクの処理を行う演算処理手段と、

前記レジスタセット群に含まれる任意のレジスタセットの内容を記憶可能な外

部レジスタセット記憶手段と、

前記タスクが利用するレジスタセットの内容を、前記外部レジスタセット記憶手段に記憶している場合には、

前記レジスタセット群に含まれる、中断中か中断すべきタスクが利用するレジスタセットの内容を前記外部レジスタセット記憶手段に転送するとともに、前記外部レジスタセット記憶手段に記憶した前記タスクが利用するレジスタセットの内容を、前記レジスタセット群のうちの前記外部レジスタセット記憶手段に転送したレジスタセットに対応するレジスタセットに転送する記憶制御手段と、を備えることを特徴とする請求項 1 に記載のプロセッサ。

【請求項 9】

前記実行タスク決定手段により決定されたタスクが利用する前記レジスタセットが前記レジスタセット群の中に存在するか否かを判定するヒット判定手段を備え、

前記ヒット判定手段が、前記タスクが利用するレジスタセットが前記レジスタセット群に存在すると判定した場合には、

前記レジスタセット選択手段は、前記レジスタセット群のうちの前記レジスタセットを選択し、

他方、存在しないと判断した場合には、

前記記憶制御手段が前記外部レジスタセット記憶手段に記憶している前記タスクが利用するレジスタセットの内容を、前記レジスタセット群のうちの転送先に対応するレジスタセットに転送した後に、前記レジスタセット選択手段はこの転送後のレジスタセットを選択することを特徴とする請求項 8 に記載のプロセッサ。

【請求項 10】

前記ヒット判定手段により存在しないと判定したときに、前記外部レジスタセット記憶手段に待避させるべき前記レジスタセット群のレジスタセットを選択する待避レジスタセット決定手段を備え、

前記記憶制御手段は、前記待避レジスタセット決定手段により選択されたレジスタセットの内容を前記外部レジスタセット記憶手段に転送することを特徴とす

る請求項 9 に記載のプロセッサ。

【請求項 11】

前記実行タスク決定手段は、前記タスク判断手段による判断結果を考慮に入れるとともに、前記記憶制御手段が前記レジスタセット群と前記外部レジスタセット記憶手段との間でレジスタセットの入れ替え処理を行っている最中にも、他のタスクの処理ができるように該入れ替え処理の影響を受けずに実行可能なタスクを決定することを特徴とする請求項 8 に記載のプロセッサ。

【請求項 12】

前記記憶制御手段は、前記外部レジスタセット記憶手段の記憶内容を更新する必要がある場合、前記レジスタセット群の変更のあったレジスタの値のみ、前記外部レジスタセット記憶手段に転送することを特徴とする請求項 8 に記載のプロセッサ。

【請求項 13】

前記記憶制御手段は、前記外部レジスタセット記憶手段から前記レジスタセット群にデータを転送する必要がある場合には、他のタスクの実行によって前記レジスタセット群のレジスタの値が書き換えられてしまったレジスタの内容のみ、前記外部レジスタセット記憶手段から前記レジスタセット群に転送することを特徴とする請求項 8 のいずれかに記載のプロセッサ。

【請求項 14】

前記記憶制御手段は、前記外部レジスタセット記憶手段に記憶されているレジスタセットと同じレジスタセットが前記レジスタセット群に存在しない場合に限り、該レジスタセットの内容を前記外部レジスタセット記憶手段に転送することを特徴とする請求項 8 に記載のプロセッサ。

【請求項 15】

前記外部レジスタセット記憶手段は、変更のあったレジスタの値及び該レジスタの識別情報だけを記憶することを特徴とする請求項 8 に記載のプロセッサ。

【請求項 16】

複数のタスクを時分割で処理するプロセッサの、前記複数のタスクの中から次に起動すべきタスクを決定する実行タスク決定装置において、

前記複数のタスクのそれぞれについて、該タスクの処理の入力データを記憶するための 1 以上の第 1 の FIFO 型記憶手段に、前記タスクが処理可能なデータが記憶されているか否か、および前記タスクの処理結果を記憶するための 1 以上の第 2 の FIFO 型記憶手段に、該処理結果のデータを記憶できる空き領域があるか否かを判断し、この判断に基づいて前記複数のタスクの中から次に起動すべきタスクを決定する実行タスク決定手段を備えることを特徴とする実行タスク決定装置。

【請求項 17】

複数のタスクを時分割で処理する演算処理方法において、

前記複数のタスクのそれぞれについて、該タスクの処理の入力データを記憶するための 1 以上の第 1 の FIFO 型記憶手段に、前記タスクが処理可能なデータが記憶されているか否か、および前記タスクの処理結果を記憶するための 1 以上の第 2 の FIFO 型記憶手段に、該処理結果のデータを記憶できる空き領域があるか否かを判断し、この判断に基づいて前記複数のタスクの中から次に起動すべきタスクを決定することを特徴とする演算処理方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は、複数のタスクを時分割で実行するプロセッサ及び実行タスク決定装置に関する。

【0002】

【従来の技術】

プロセッサが時分割で複数のタスクを実行する場合、オペレーティングシステムなどのソフトウェアを用いてタスク切り替えを行うのが一般的である。

【0003】

オペレーティングシステムは、一定時間ごとに、各タスクが実行可能になったかどうかを検出する。実行可能の可否は、たとえば処理対象となるデータが準備できているかで判断できる。

【0004】

このような判断処理をオペレーティングシステムが行っている間は、タスクの

実行を中断しなければならない。タスクの実行を中断する際には、汎用レジスタ、スタックポインタ (SP) 及びプログラムカウンタ (PC) などの内容がプロセッサ外部の主記憶メモリなどに退避される。退避された各種の情報は タスクの識別情報 (タスク ID) とともにオペレーティングシステムが管理する。

【0005】

オペレーティングシステムはスケジューラを持っており、このスケジューラにより、たとえばラウンドロビンなどの方式に従って、実行可能なタスクの中から実際に実行するタスクを決定する。

【0006】

オペレーティングシステムは、スケジューラにより実行が決定されたタスクについての、主記憶メモリ等に退避された各種情報を取得し、汎用レジスタ、スタックポインタ (SP) 及びプログラムカウンタ (PC) などの内容を退避先のプロセッサ外の主記憶装置などから各レジスタに復元し、その後タスクの実行を開始する。

【0007】

ハードウェアによる複数のデータ処理ユニットを用いてリアルタイムデータの並列処理を実現する方法も提案されている (たとえば特許文献 1 を参照)。しかしながら並列処理を行うためにデータ処理コアを複数設け、さらにそれらの調停機構を設けるなど、構成自体が複雑となる問題点がある。

【0008】

【特許文献 1】

特表 2000-509528 公報

【0009】

【発明が解決しようとする課題】

オペレーティングシステムなどのソフトウェアを用いたタスク切り替えでは、オペレーティングシステム自体の動作によるオーバーヘッドがあるため、本来のタスク処理に必要な性能よりも高い性能のプロセッサを用意する必要があった。

【0010】

また、オペレーティングシステムによるタスク切り替えでは、タスク切り替え

に時間がかかり、リアルタイム性のあるシステムの構築が困難であった。

【0011】

さらにオペレーティングシステムは、タスクが実行可能かどうかの判断を一定時間ごとにしか行わないため、タスクが実行可能になってからオペレーティングシステムに実行可能と判断され、実際にタスクが実行されるまでに時間がかかることがあり、応答性が悪かった。この応答性の悪さがリアルタイム性のあるシステムの構築をさらに困難にしていた。

【0012】

また、処理対象となるデータの準備が出来たかどうかを条件としてタスク起動可能を判定する場合、処理結果を格納する領域にデータを格納出来ない場合があり、そのような場合はデータを失うことになる。

【0013】

一方、多くをハードウェア化しリアルタイム処理を実現しようとする、従来は構成が複雑かつコスト高になりやすい傾向があった。

【0014】

本発明は、このような点に鑑みてなされたものであり、その目的は、複数のタスクをリアルタイムに効率よく実行できるプロセッサおよび実行タスク決定装置を提供することにある。

【0015】

【課題を解決するための手段】

上述した課題を解決するために、本発明は、複数のタスクを時分割で処理するプロセッサにおいて、予め定めたタスクに対して、該タスクの処理で使用されるデータを記憶する1以上の第1のFIFO (First In First Out) 型記憶手段と、前記第1のFIFO型記憶手段から取得したデータを使用して前記タスクの処理を行った後の処理結果を記憶するとともに、この記憶した処理結果を使用する他のタスクがある場合には該他のタスクの処理で使用されるデータを記憶する1以上の第2のFIFO型記憶手段と、前記第1のFIFO型記憶手段が前記タスクの処理に使用するデータを保持しているか否か、および前記タスクの処理結果を記憶する空き領域を前記第2のFIFO型記憶手段が有しているか否かを判断するタスク判断手段と

、前記タスク判断手段の判断結果に基づいて、前記複数のタスクのうちから次に起動すべきタスクを決定する実行タスク決定手段と、を備える。

【0016】

【発明の実施の形態】

以下、本発明に係るプロセッサ及び実行タスク決定装置について、図面を参照しながら具体的に説明する。

【0017】

(本発明の原理)

複数のタスクを時分割でリアルタイムに実行したいという要求が高まっている。例えば、図1はソフトウェア無線機の処理内容を示す図である。図1のソフトウェア無線機では、処理A～処理Dを連続的に実行する。例えば、処理Bは、処理Aの出力を入力として所定の演算処理を行った結果を処理Cに渡す。

【0018】

図1において、各処理での出力データ数とその次の処理を起動するのに必要なデータ数とは必ずしも一致しない。また、各処理の処理量も処理内容によってさまざまである。さらに、処理間でデータ受け渡しのタイミングが異なるため、各処理の段間で同期を取る必要がある。

【0019】

図1の処理を効率よく実現するために、各処理を1つ以上のタスクとして生成することが多い。各タスクは、ソフトウェアで実現される場合とハードウェアで実現される場合がある。

【0020】

本発明では、タスク同士の厳密な同期を取らなくても済むように、タスク間にFIFO (First In First Out) を挿入する。FIFOとは一般にはデータの格納方法を指すが、以下に示す実施形態では格納したデータが過去に格納したものから順に取り出される方式、つまりFIFO型の記憶装置を指している。一番新しく格納されたデータがもっとも最後に取り出されるもので、たとえばキューと呼ばれるデータ構造はこの方式の記憶装置である。この場合のプロセッサの概略構成は図2のようになる。

【0021】

図2において、FIFO1はタスク1への入力データを出力し、FIFO2はタスク1からの出力データを受け取るとともに、タスク2への入力データを出力する。

【0022】

図2では、各タスクの入力側及び出力側に1つずつFIFOを設けているが、複数のFIFOを設けてもよい。

【0023】

ソフトウェア無線機を図2の構成で実現することも可能であるが、各タスク同士で同期を取るための処理量が多いことから、FIFOを利用したとしても、ソフトウェアで同期を取るのは、プロセッサのオーバーヘッドが大きくなり、実現が困難である。

【0024】

ここで、各タスクの起動条件に着目すると、各タスクは入力データが存在しなときには起動する必要はない。また、各タスクの出力側のFIFOに各タスクの処理結果を格納する空き領域が存在しない場合には、各タスクの処理結果を失うおそれがあり、この場合もタスクを起動する必要はない。

【0025】

そこで、以下に説明する本発明の各実施形態は、FIFO内のデータの格納状態を監視しておき、各タスクへの入力データが存在し、かつ出力側のFIFOに空き領域が存在する場合のみ、タスクを起動している。

【0026】

(第1の実施形態)

図3は本発明に係るプロセッサの第1の実施形態の概略構成を示すブロック図である。図3のプロセッサは、複数のタスクそれぞれが起動可能か否かを判断するタスク起動部11と、タスク起動部11の判断結果に基づいて起動すべきタスクを決定する実行タスク決定部12と、実行タスク決定部12で決定されたタスクを実行するプロセッサコア13とを備えている。

【0027】

タスク起動部11には、複数のFIFO10が接続されている。各FIFO10は、図

2のFIFO10に対応するものであり、タスク起動部11は各FIFO10間でデータの受け渡しを行う。より具体的には、タスク起動部11は、タスクに入力されるべきFIFO10の出力データを保持するとともに、タスクの処理結果を格納すべき他のFIFO10に空き領域が存在するか否かを確認する。そして、タスクに入力されるべきFIFO10の出力データが存在し、かつタスクの処理結果を格納すべき他のFIFO10に空き領域が存在する場合に限り、タスクを起動すべきと判断する。

【0028】

実行タスク決定部12は、起動可能なタスクが複数存在する場合には、その中からいずれか1つのタスクを選択する。例えば、各タスクを識別するためのタスクIDが最小のタスクを実行する。

【0029】

プロセッサコア13は、例えば、命令取得部、命令解釈部、命令実行部、命令実行結果を格納する命令実行結果格納部、及び命令の実行状態を格納する命令実行状態格納部を有する。

【0030】

このように、第1の実施形態では、タスクに入力されるべきデータをFIFO10が保持し、かつタスクの実行結果を格納すべき他のFIFO10に空き領域が存在するか否かをタスク起動部11で判断し、その判断結果に基づいて実行タスク決定部12が起動すべきタスクを決定するようにしたため、オペレーティングシステムなしでタスクのスケジューリングを行うことができ、オペレーティングシステムのオーバーヘッドの影響を受けなくなり、タスクのリアルタイム処理が可能になる。

【0031】

(第2の実施形態)

第2の実施形態は、複数のタスクに優先順位をつけて、優先順位に従ってタスクの実行を行うものである。

【0032】

図4は本発明に係るプロセッサの第2の実施形態の概略構成を示すブロック図である。図4のプロセッサは、図3の構成に加えて、複数のタスクそれぞれのタ

タスクIDと優先順位を供給するタスクID・優先順位供給部14を備えている。

【0033】

図5は実行タスク決定部12とタスクID・優先順位供給部14の詳細構成を示すブロック図である。図示のように、タスクID・優先順位供給部14は、各タスクのタスクIDと優先順位との対応関係を記憶している。実行タスク決定部12は、タスク起動部11から通知された実行可能なタスクの一覧を登録した実行可能タスクIDリスト15と、起動すべきタスクを決定するプライオリティ・エンコーダ16とを有する。

【0034】

プライオリティ・エンコーダ16は、タスクIDリストに登録されているタスクの優先順位をタスクID・優先順位供給部14から取得し、例えば優先順位が最も高いタスクを起動タスクとして決定する。

【0035】

タスクID・優先順位供給部14が供給する優先順位の設定方法については特に制限はないが、例えば、新たなタスクを生成するときに、そのタスクの優先順位を設定してもよいし、事後的に優先順位を変更できるようにしてもよい。

【0036】

図6は図5の実行タスク決定部12の処理手順を示すフローチャートである。まず、実行可能タスクIDリスト15に登録されているタスクのタスクIDを参照し（ステップS1）、そのタスクIDに対応する優先順位をタスクID・優先順位供給部14から取得する（ステップS2）。

【0037】

次に、ステップS2で取得した優先順位が、タスクの起動候補の優先順位より高いか否かを判定し（ステップS3）、高ければ、ステップS1で参照したタスクIDをもつタスクを起動候補として設定する（ステップS4）。

【0038】

ステップS4の処理が終了した場合、またはステップS3で高くないと判定された場合には、まだ優先順位を比較していないタスクが実行可能タスクIDリスト15に残っているか否かを判定し（ステップS5）、残っている場合にはステッ

プ S 1 以降の処理を繰返し、残っていない場合には処理を終了する。

【0039】

このように、第 2 の実施形態では、タスクの優先順位を予め定めておくため、複数のタスクが起動候補として挙げられた場合に、優先順位の高いタスクを優先的に実行でき、処理の効率化が図れる。また、起動すべきタスクの選択を迅速に行うこともできる。

【0040】

(第 3 の実施形態)

第 3 の実施形態は、動的に優先順位を変更できるようにしたものである。

【0041】

図 7 は本発明に係るプロセッサの第 3 の実施形態の概略構成を示すブロック図である。図 7 のプロセッサは、図 4 の構成に加えて、タスクの優先順位を動的に変更する優先順位変更部 30 を備えている。

【0042】

優先順位変更部 30 は、タスクごとの実行時刻を計測する時刻計測部 17 と、各タスクのタスク ID と過去の起動時刻との対応関係を登録した起動時刻テーブル 18 と、各タスクの起動間隔の平均を計算する平均時間間隔計算部 19 と、平均時間間隔計算部 19 での計算結果に基づいて各タスクの優先順位を決定する優先順位決定部 20 とを有する。

【0043】

平均時間間隔計算部 19 の計算結果が小さいほど、より頻繁に起動されるタスクであることを示している。このため、優先順位決定部 20 は、例えば、平均時間間隔計算部 19 の計算結果が小さいタスクほど、優先順位を高くする。

【0044】

より具体的には、平均時間間隔計算部 19 は各 FIFO 10 へのデータ入力間隔を算出し、その間隔の短い FIFO 10 に対応するタスクの優先順位が高くなるように、優先順位決定部 20 で優先順位を決定する。あるいは、平均時間間隔計算部 19 はプロセッサコア 13 から通知される各タスクの実行間隔を算出し、その間隔の短いタスクの優先順位が高くなるように、優先順位決定部 20 で優先順位を決

定してもよい。あるいは、平均時間間隔計算部 19 は各タスクにデータを供給する各 FIFO 10 の空き領域を監視し、空き領域の少ない FIFO 10 に対応するタスクの優先順位が高くなるように、優先順位決定部 20 で優先順位を決定してもよい。あるいは、平均時間間隔計算部 19 は各タスクから各 FIFO 10 へのデータ出力間隔を算出し、その間隔の短い FIFO 10 に対応するタスクの優先順位が高くなるように、優先順位決定部 20 で優先順位を決定してもよい。

【0045】

このように、第 3 の実施形態では、各タスクの優先順位を動的に変更できるようにしたため、過去の実行履歴を参考にしてタスクのスケジューリングを行うことができる。

【0046】

(第 4 の実施形態)

第 4 の実施形態は、実行中のタスクよりも優先順位の高いタスクの起動準備が整うと、実行中のタスクを中断して、優先順位の高いタスクを起動するものである。

【0047】

第 4 の実施形態のブロック構成は、図 4 や図 7 と同様である。図 8 は本発明に係るプロセッサの第 4 の実施形態の動作を説明するタイムチャートである。図 8 は 3 つのタスク A, B, C が存在し、タスク A の優先順位が最も高く、次にタスク B の優先順位が高く、タスク C の優先順位が最も低い例を示している。

【0048】

まず、時刻 t_0 でタスク C が起動したとする。その後、時刻 t_1 でタスク B の起動準備が整ったとする。これにより、実行タスク決定部 12 はタスク C の実行を中断して、代わりにタスク B の起動を開始する。

【0049】

その後、時刻 t_2 でタスク A の起動準備が整ったとする。これにより、実行タスク決定部 12 はタスク B の実行を中断して、代わりにタスク A の起動を開始する。

【0050】

その後、時刻 t_3 でタスク A の実行が終了したとすると、次に優先順位の高いタスク B の実行を再開し、時刻 t_4 でタスク B の実行が終了したとすると、次に一番優先順位の低いタスク C の実行を開始する。

【0051】

このように、第 4 の実施形態では、実行中のタスクよりも優先順位の高いタスクの起動準備が整うと、実行中のタスクの処理を中断して、優先順位の高いタスクの実行を開始するようにしたため、重要なタスクを常に優先的に処理でき、プロセッサ全体の処理能力の向上が図れる。

【0052】

(第 5 の実施形態)

以下に説明する第 5 ～ 第 11 の実施形態は、実行中のタスクを迅速に切り替える仕組みに関する。

【0053】

図 9 は本発明に係るプロセッサの第 5 の実施形態の概略構成を示すブロック図である。図 9 のプロセッサは、図 4 と同様のタスク起動部 11、実行タスク決定部 12 及びタスク ID・優先順位供給部 14 を備えており、その他に、複数のレジスタセットからなるレジスタセット群 21 と、レジスタセット群 21 の中からいずれか 1 つのレジスタセットを選択するレジスタセット選択部 22 と、選択されたレジスタセットを用いて演算処理を行う演算部 23 と、を備えている。これらレジスタセット、レジスタセット選択部 22 及び演算部 23 でプロセッサコア 13 を構成している。

【0054】

レジスタセット選択部 22 の内部には、レジスタセット群 21 の中から 1 つのレジスタセットを選択するデコーダ 22a が設けられている。このデコーダ 22a は、実行タスク決定部 12 からのタスク ID に対応するレジスタセットの選択信号を出力する。

【0055】

各レジスタセットは、個々のタスクに固有の情報をすべて含む 1 以上のレジスタからなる。レジスタセットを構成する各レジスタの種類は、プロセッサのアー

キテクチャに依存し、例えば、プログラムカウンタ (PC)、スタックポインタ (SP)、または汎用レジスタ (R0, R1, ...) などである。本明細書では、1つのレジスタセット内のレジスタの総数を r (r は 1 以上の整数) とする。

【0056】

レジスタセットの総数は、時分割で実行可能なタスクの数 n 以上の m 個 (m は整数) である。各レジスタセットは、固有の識別番号 (レジスタセット ID) で識別される。

【0057】

実行中のタスクは、1つのレジスタセットを用いる。タスクごとに使用するレジスタの種類が異なる場合には、各タスクごとに専用のレジスタセットを設けてもよい。プロセッサの設計時に、レジスタセットを構成するレジスタの種類を決めてもよいし、プログラム命令により事後的にレジスタセットを構成するレジスタの種類を変更してもよい。

【0058】

タスクごとに使用するレジスタの種類が異なる場合には、必要に応じて、図 10 に示すようなタスク ID とレジスタセット ID との対応関係を登録したレジスタセット ID タスク ID 対応部 24 を設け、このテーブルを参照して、各タスクに対応するレジスタセットを決定してもよい。

【0059】

レジスタセット ID タスク ID 対応部 24 は、より詳細には、図 11 のようなテーブルで構成される。図 11 のテーブルは、プロセッサの設計時に作成してその後の変更ができないようにしてもよいし、プロセッサの起動後に何らかの命令によりテーブルの内容を変更できるようにしてもよい。

【0060】

図 9 や図 10 の実行タスク決定部 12 がタスクの切替を行うと、切替後のタスクのタスク ID が実行タスク決定部 12 からレジスタセット選択部 22 に通知される。レジスタセット選択部 22 は、図 11 のテーブル等を用いて、切替後のタスクに対応するレジスタセット ID を取得し、その ID に対応するレジスタセットの値を演算部 23 に供給する。演算部 23 は、レジスタセット選択部 22 で選択され

たレジスタセットの値を各レジスタに設定して演算処理を行い、その演算処理結果をレジスタセット選択部 22 で選択されたレジスタセットに格納する。

【0061】

このように、第5の実施形態では、タスクの切替が起きると、レジスタセットも切り替えるようにしたため、タスク切替時の処理準備時間を短縮でき、タスクの高速切替を実現できる。また、いったんタスクの処理を中断した後に再開する場合も、再開前にレジスタセットの値を読み込めばよいため、いったん中断したタスクの処理を迅速に再開することができる。

【0062】

(第6の実施形態)

第6の実施形態は、レジスタセット群 21 の中の少なくとも一部のレジスタセットを外部に退避するものである。

【0063】

図 12 は本発明に係るプロセッサの第6の実施形態の概略構成を示すブロック図である。図 12 のプロセッサは、図 10 の構成に加えて、ヒット判定部 31 と、外部レジスタセット記憶部 32 と、外部記憶制御部 33 と、レジスタセット転送部 34 と、を有する。

【0064】

本実施形態のレジスタセット群 21 は、時分割で実行するタスクの数 n より多くても少なくとも等しくても構わない。

【0065】

ヒット判定部 31 は、実行すべきタスクのレジスタセットがレジスタセット ID タスク ID 対応部 24 に登録されているか否かを判定する。レジスタセット転送部 34 は、レジスタセット群 21 の中の少なくとも一部のレジスタセットの内容を外部レジスタセット記憶部 32 に転送するとともに、外部レジスタセット記憶部 32 から読み出したレジスタセットの内容をレジスタセット群 21 に転送する。外部レジスタセット記憶部 32 は、レジスタセット群 21 の中の少なくとも一部のレジスタセットの内容を記憶する。

【0066】

図13は外部レジスタセット記憶部32のデータ構成を示す図である。外部レジスタセット記憶部32は、レジスタセット群21に含まれる任意のレジスタセットの内容を記憶でき、また記憶したレジスタセットの内容をレジスタセット群21に転送することもできる。外部レジスタセット記憶部32に記憶された各レジスタセットはタスクIDで管理され、いったん記憶されたレジスタセットの内容を呼び出す際も、タスクIDを指定する。

【0067】

このように、外部レジスタセット記憶部32は、レジスタセット群21の中の少なくとも一部のレジスタセットの内容を一時的に退避し、必要に応じてレジスタセット群21に転送することができる。

【0068】

外部レジスタセット記憶部32は、専用のハードウェアで構成してもよいし、主記憶メモリなどの予め設けられているメモリの一部領域を利用してもよい。

【0069】

図14はヒット判定部31の処理手順を示すフローチャートである。まず、実行すべきタスクを表すタスクIDに対応するレジスタセットIDがレジスタセットIDタスクID対応部24に登録されているか否かを判定する（ステップS11）。登録されていれば、ヒットしたと判定して、該当するタスクIDに対応するレジスタセットIDを取得する（ステップS12）。この場合、図5と同様の手順でタスク切替を行う。

【0070】

一方、登録されていなければ、ミスしたと判定し（ステップS13）、外部レジスタセット記憶部32から該当するレジスタセットを呼び出して、レジスタセット群21の一部の入れ替え処理を行う。

【0071】

より具体的には、まず、レジスタセット群21の中の少なくとも一部のレジスタセットの内容をレジスタセット転送部34を介して外部レジスタセット記憶部32に転送する。同時に、実行すべきタスクが利用するレジスタセットの内容を外部レジスタセット記憶部32から読み出して、レジスタセット転送部34を介

してレジスタセット群 21 に転送する。

【0072】

このように、第 6 の実施形態では、レジスタセット群 21 の中の少なくとも一部のレジスタセットの内容を外部レジスタセット記憶部 32 に退避し、必要に応じて外部レジスタセット記憶部 32 からレジスタセット群 21 にレジスタセットの内容を戻せるようにしたため、レジスタセット群 21 の中のレジスタセットの総数以上のタスクを処理することができる。したがって、レジスタセット群 21 の中のレジスタセットの数を削減でき、プロセッサのサイズを小型化できる。

【0073】

(第 7 の実施形態)

第 7 の実施形態は、ヒット判定部 31 でヒットしなかったときに、退避させるべきレジスタセットを予め指定するものである。

【0074】

図 15 は本発明に係るプロセッサの第 7 の実施形態の概略構成を示すブロック図である。図 15 のプロセッサは、図 12 の構成に加えて、退避レジスタセット決定部 35 を備えている。

【0075】

退避レジスタセット決定部 35 は、タスク ID・優先順位供給部 14 から供給された優先順位に基づいて、外部レジスタセット記憶部 32 に退避させるべきレジスタセットをレジスタセット群 21 の中から決定する。レジスタセット転送部 34 は、退避レジスタセット決定部 35 で決定されたレジスタセットの内容を外部レジスタセット記憶部 32 に転送するとともに、外部レジスタセット記憶部 32 から読み出したレジスタセットの内容をレジスタセット群 21 に転送する。

【0076】

図 16 は退避レジスタセット決定部 35 の処理手順を示すフローチャートである。まず、レジスタセット群 21 の中に利用するレジスタセットが存在し、かつそのレジスタセットに対応するタスクが実行不可能であるか否かを判定する（ステップ S21）。判定が Yes の場合、該当するタスクの中で最も優先順位の低いタスクを選択する（ステップ S22）。

【0077】

一方、ステップS21の判定がNoの場合、レジスタセット群21の中の各レジスタセットに対応するタスクのうち、最も優先順位の低いタスクを選択する（ステップS23）。

【0078】

ステップS22またはS23の処理が終了すると、選択されたタスクが利用するレジスタセットのID（レジスタセットID）を取得し（ステップS24）、取得したIDをレジスタセット転送部34に指示する（ステップS25）。

【0079】

このように、第7の実施形態では、外部レジスタセット記憶部32に退避すべきレジスタセットを明示的に指定できるようにしたため、使用頻度の低いレジスタセットを入れ替えることができ、レジスタセットの退避による処理効率の低下を防止できる。

【0080】

（第8の実施形態）

上述した第6及び第7の実施形態は、実行すべきタスクが利用するレジスタセットがレジスタセット群21に存在するか否かをヒット判定部31で判定する処理を行うため、タスクの切替を迅速に行えないおそれがある。そこで、以下に説明する第8の実施形態は、ヒット判定部31が処理を行っている間に、ヒット判定部31の判定結果の影響を受けないタスクの処理を行うものである。

【0081】

図17は本発明に係るプロセッサの第8の実施形態の概略構成を示すブロック図である。図17の実行タスク決定部12は、図15の実行タスク決定部12とは異なる処理を行う。すなわち、図17の実行タスク決定部12は、ヒット判定部31に対して判定結果の送信を要求し、ヒット判定部31からの判定結果を受信する。

【0082】

図17のレジスタセットIDタスクID対応部24は、図18に示すように、個々のタスクID及びレジスタセットIDに対応づけて、フラグを有する。このフラグは

、あるタスクを実行しようとしたが、ヒット判定部 31 でミスし、そのタスクが利用するレジスタセットの内容を外部レジスタセット記憶部 32 から転送する処理を行っている最中にセットされる。

【0083】

実行タスク決定部 12 は、実行するタスクを切り替える際に、ミス時転送ヒット判定要求またはヒット判定要求のいずれかを出す。ミス時転送ヒット判定要求は、ヒット判定部 31 に対してタスク ID を通知し、ミスした場合には、そのタスクが利用するレジスタセットを外部レジスタセット転送装置からレジスタセット群 21 に転送するようレジスタセット選択部 22 に指示するものである。

【0084】

ミス時転送ヒット判定要求を受けた場合のヒット判定部 31 の処理手順は図 19 のフローチャートで表される。まず、実行すべきタスクのタスク ID がレジスタセット ID タスク ID 対応部 24 に登録されているか否かを判定する（ステップ S31）。登録されていれば、該当するタスク ID に対応するレジスタセット ID を取得し（ステップ S32）、登録されていなければ、ミスと判定して（ステップ S33）、該当するタスクが利用するレジスタセットの転送をレジスタセット転送部 34 に指示する（ステップ S34）。

【0085】

一方、ヒット判定要求は、ヒット判定部 31 に対してタスク ID を通知し、フラグが有効であるレジスタセットを除外してヒット判定を行うものである。ミスした場合でもレジスタセットの転送は行わない。

【0086】

ヒット判定要求を受けた場合のヒット判定部 31 の処理手順は図 20 のフローチャートで表される。まず、フラグがセットされているタスク ID を除いて、実行すべきタスクのタスク ID がレジスタセット ID タスク ID 対応部 24 に登録されているか否かを判定する（ステップ S41）。登録されていれば、そのタスク ID に対応するレジスタセット ID を取得し（ステップ S42）、登録されていなければ、ミスしたと判定する（ステップ S43）。

【0087】

レジスタセットIDタスクID対応部24の中に、セットされているフラグが存在しない場合には、実行タスク決定部12は、実行するタスクを切り替える際に、切替後のタスクIDでミス時転送ヒット判定要求を出す。ヒットした場合には、実行タスク決定部12は、ヒットした新しいタスクIDに対応するレジスタセットIDをレジスタセットIDタスクID対応部24から取得し、そのレジスタセットIDをレジスタセット選択部22に通知し、新しいタスクの実行を開始する。一方、ミスした場合は、ヒット判定部31はレジスタセット転送部34にレジスタセットの転送を指示し、レジスタセットIDタスクID対応部24は、転送を行うレジスタセットに対応するフラグをセットする。

【0088】

レジスタセットの転送が終了すると、レジスタセット転送部34は、レジスタセットIDタスクID対応部24に転送完了を通知する。この通知に従って、レジスタセットIDタスクID対応部24はレジスタセットのフラグを無効化する。

【0089】

実行タスク決定部12は、実行すべきタスクに対応するレジスタセットIDをレジスタセットIDタスクID対応部24から取得し、取得したレジスタセットIDをレジスタセット選択部22に通知して、そのタスクの実行を開始する。

【0090】

レジスタセットIDタスクID対応部24の中に、セットされているフラグが存在する場合には、実行タスク決定部12は、実行するタスクを切り替える際に、切替後のタスクIDでヒット判定要求を出す。ヒットした場合には、実行タスク決定部12は、ヒットした新しいタスクIDに対応するレジスタセットIDをレジスタセットIDタスクID対応部24から取得し、そのレジスタセットIDをレジスタセット選択部22に通知し、新しいタスクの実行を開始する。一方、ミスした場合には、タスクの切替を行わない。

【0091】

図21はレジスタセットIDタスクID対応部24の中にセットされているフラグが存在する場合のタスクの実行形態を説明する図である。この例では、タスクID5のタスクを実行中に、このタスクよりも優先順位の高いタスクID1のタスクの実

行が可能になったときに、実行タスク決定部 12 はヒット判定部 31 に対してタスク ID1 のミス時転送ヒット判定要求を行う。

【0092】

この例では、この要求を受けたヒット判定部 31 はミス判定を行っており、これにより、レジスタセット転送部 34 はタスク ID1 が利用するレジスタをレジスタセット群 21 で転送を開始する。この転送には、時刻 t_3 までの時間がかかる。

【0093】

時刻 $t_1 \sim t_3$ の間の時刻 t_2 でタスク ID5 の実行が終了すると、実行タスク決定分は実行可能なタスクの中で最も優先順位の高いタスク（図 21 の例ではタスク ID7）のヒット判定要求を行う。ヒットした場合には、タスク ID7 の実行を行う。

【0094】

その後、時刻 t_3 になると、レジスタセット転送部 34 からの転送完了が通知され、それによって実行タスク決定分はタスク ID1 の実行を開始する。

【0095】

このように、第 8 の実施形態では、あるタスクが利用するレジスタセットの転送中に、そのタスクの実行に影響しない他のタスクの実行を行うようにしたため、レジスタセットの転送完了を待つことなくタスクの処理を行え、タスクの処理効率を向上できる。

【0096】

（第 9 の実施形態）

第 9 の実施形態は、レジスタセットを構成する各レジスタに、レジスタの内容が更新されたか否かを示すフラグを設けるものである。

【0097】

第 9 の実施形態は、図 12 と同様のブロック構成を有するが、レジスタセット群 21 のレジスタセットのデータ構造が図 12 とは異なっている。図 22 は第 9 の実施形態のレジスタセットのデータ構造を示す図である。図示のように、レジスタセットを構成する各レジスタに対応して変更フラグが設けられている。この

変更フラグは、対応するレジスタの内容を書き換えるときにセットされる。この場合の処理手順は図 23 のようなフローチャートで表される。

【0098】

図 23 において、まず、書き換えるレジスタに対応する変更フラグをセットし（ステップ S51）、その後にレジスタを書き換える（ステップ S52）。

【0099】

図 22 の変更フラグは、外部レジスタセット記憶部 32 に記憶されているレジスタセットを読み込むときにクリアされる。外部レジスタセット記憶部 32 からレジスタセット群 21 への転送要求があった場合の処理手順は図 24 のようなフローチャートで表される。

【0100】

図 24 において、まず、転送先のレジスタセット内の変更フラグをすべてクリアし（ステップ S61）、その後にレジスタセットを転送する（ステップ S62）。

【0101】

図 25 はレジスタセット群 21 が外部レジスタセット記憶部 32 にレジスタセットの内容の転送要求を行った場合のレジスタセット転送部 34 の処理手順を示すフローチャートである。まず、転送するレジスタセット内の 1 つのレジスタについて注目し（ステップ S71）、注目したレジスタセットの変更フラグがセットされているか否かを判定する（ステップ S72）。

【0102】

セットされている場合には、転送するレジスタを外部レジスタセット記憶部 32 に転送する（ステップ S73）。セットされていない場合か、ステップ S73 の処理が終了した場合は、すべてのレジスタについて処理を行ったか否かを判定し（ステップ S74）、処理を行っていないものがあればステップ S71 に戻り、処理を行った場合は終了する。

【0103】

このように、第 9 の実施形態では、レジスタセット群 21 の中に含まれるレジスタのうち、変更のあったものだけを外部レジスタセット記憶部 32 に転送する

ようにしたため、転送すべきデータ量を削減でき、転送時間の削減が図れる。

【0104】

(第10の実施形態)

第10の実施形態は、外部レジスタセット記憶部32の中に、各レジスタセットの各レジスタが書き換えられたか否かを示すフラグを有する。

【0105】

図26は外部レジスタセット記憶部32のデータ構造を示す図である。図示のように、各レジスタセットの各レジスタに対応して、有効フラグが設けられている。この有効フラグは、対応するレジスタが書き換えられた場合にセットされる。

【0106】

図27は有効フラグの初期化処理を示すフローチャートであり、タスクの起動時に外部記憶制御部33が行う処理である。外部記憶制御部33は、起動するタスクに対応するレジスタセット内の有効フラグすべてをクリアする(ステップS81)。

【0107】

図28は有効フラグのセット処理を示すフローチャートであり、レジスタセット転送部34から外部レジスタセット記憶部32へのレジスタセットの転送要求があったときに、外部記憶制御部33が行う処理である。外部記憶制御部33は、外部レジスタセット記憶部32の転送するレジスタに対応する有効フラグをセットする(ステップS91)。その後、レジスタセット群21から外部レジスタセット記憶部32への該当レジスタの転送を行う(ステップS92)。

【0108】

図29は外部レジスタセット記憶部32からレジスタセット群21への転送要求があった場合の外部記憶制御部33の処理手順を示すフローチャートである。まず、転送するレジスタセット内の有効フラグを読み込む(ステップS101)。次に、転送するレジスタセット内の1つのレジスタについて着目する(ステップS102)。

【0109】

このレジスタに対応する有効フラグがセットされているか否かを判定し（ステップS103）、セットされていれば、レジスタを外部レジスタセット記憶部32に転送する（ステップS104）。次に、すべてのレジスタについて処理を行ったか否かを判定し（ステップS105）、まだ処理を行っていないレジスタがあれば、ステップS101以降の処理を繰り返す。

【0110】

このように、第10の実施形態では、外部レジスタセット記憶部32の中に、レジスタの書き換えを行ったか否かを示す有効フラグを設けるため、レジスタの内容が書き換えられた場合のみ、外部レジスタセット記憶部32からレジスタセット群21にレジスタの内容を転送すればよく、転送回数を削減できるとともに、転送時間も短縮できる。

【0111】

（第11の実施形態）

第11の実施形態は、外部レジスタセット記憶部32の各タスクのレジスタセットをリスト構造にするものである。

【0112】

図30は第11の実施形態の外部レジスタセット記憶部32のデータ構造を示す図である。外部レジスタセット記憶部32は、リスト構造になっており、変更のあったレジスタの値を記憶するレジスタ値記憶領域と、レジスタ値記憶領域に記憶されているレジスタの識別番号（レジスタID）とを組にして記憶する。すなわち、外部レジスタセット記憶部32は、内容に変更のないレジスタの値は記憶しない。

【0113】

図31は外部レジスタセット記憶部32からレジスタセット群21への転送要求があった場合の外部記憶制御部33の処理手順を示すフローチャートである。まず、転送するレジスタセットのリストの先頭にポインタを移動する（ステップS111）。次に、外部レジスタセット記憶部32のリストからレジスタIDとレジスタの値を読み込む（ステップS112）。次に、レジスタIDに対応するレジスタセット群21のレジスタにレジスタの値を書き込む（ステップS113）。

次に、リストの最後か否かを判定する（ステップS114）。リストの最後であれば処理を終了し、リストの最後でなければ、リストの次の項目に移動してステップS111以降の処理を繰り返す。

【0114】

図32はレジスタセット転送部34から外部レジスタセット記憶部32にレジスタの転送要求があった場合の外部記憶制御部33の処理手順を示すフローチャートである。まず、転送するレジスタIDが外部レジスタセット記憶部32の対応するリスト中にあるか否かを判定する（ステップS121）。リスト中になければ、リストに転送するレジスタIDの項目を追加する（ステップS122）。

【0115】

ステップS121でリスト中にあると判定された場合か、ステップS122の処理が終了した場合は、リストの転送するレジスタIDに対応するレジスタ値記憶領域に、レジスタの値を書き込む（ステップS123）。

【0116】

図33はタスク終了時の外部記憶制御部33の処理手順を示すフローチャートである。外部レジスタセット記憶部32の中の終了したタスクに対応するリストの内容を破棄する（ステップS131）。

【0117】

このように、第11の実施形態では、外部レジスタセット記憶部32をリスト構造にし、変更のあったレジスタの値のみを記憶するようにしたため、外部レジスタセット記憶部32のメモリ容量を削減できるとともに、レジスタセット群21との間で転送されるデータ量も削減できる。

【0118】

上述した第1～第11の実施形態では、本発明をプロセッサに適用した例を説明したが、本発明はプロセッサ以外の半導体集積装置にも適用可能である。例えば、図1等の実行タスク決定部12のみをプロセッサとは別個に設けてもよい。

【0119】

【発明の効果】

以上詳細に説明したように、本発明によれば、第1及び第2のFIFO型記憶手段

のデータ格納状態によりタスクの起動準備が整ったか否かを判断し、起動準備が整ったタスクのみ起動するようにしたため、オペレーティングシステムなしでタスクの起動スケジューリングを行うことができ、オペレーティングシステムのオーバーヘッドがなくなって、複数のタスクを効率よく処理することができる。

【0120】

また、タスクで処理するデータを格納する第1及び第2のFIFO型記憶手段を監視するため、データを消失するおそれがなくなり、信頼性の高いタスク処理を行うことができる。

【図面の簡単な説明】

【図1】

ソフトウェア無線機の処理内容を示す図。

【図2】

タスク間にFIFOを設けたプロセッサの概略構成図。

【図3】

本発明に係るプロセッサの第1の実施形態の概略構成を示すブロック図。

【図4】

本発明に係るプロセッサの第2の実施形態の概略構成を示すブロック図。

【図5】

実行タスク決定部12とタスクID・優先順位供給部14の詳細構成を示すブロック図。

【図6】

図5の実行タスク決定部12の処理手順を示すフローチャート。

【図7】

本発明に係るプロセッサの第3の実施形態の概略構成を示すブロック図。

【図8】

本発明に係るプロセッサの第4の実施形態の動作を説明するタイムチャート。

【図9】

本発明に係るプロセッサの第5の実施形態の概略構成を示すブロック図。

【図10】

図9の変形例を示すプロセッサのブロック図。

【図11】

レジスタセットIDタスクID対応部24のデータ構造を示す図。

【図12】

本発明に係るプロセッサの第6の実施形態の概略構成を示すブロック図。

【図13】

外部レジスタセット記憶部32のデータ構成を示す図。

【図14】

ヒット判定部31の処理手順を示すフローチャート。

【図15】

本発明に係るプロセッサの第7の実施形態の概略構成を示すブロック図。

【図16】

回避レジスタセット決定部35の処理手順を示すフローチャート。

【図17】

本発明に係るプロセッサの第8の実施形態の概略構成を示すブロック図。

【図18】

第8の実施形態のレジスタセットIDタスクID対応部のデータ構造を示す図。

【図19】

ミス時転送ヒット判定要求を受けた場合のヒット判定部の処理手順を示すフローチャート。

【図20】

ヒット判定要求を受けた場合のヒット判定部の処理手順を示すフローチャート。

【図21】

レジスタセットIDタスクID対応部24の中にセットされているフラグが存在する場合のタスクの実行形態を説明する図。

【図22】

第9の実施形態のレジスタセットのデータ構造を示す図。

【図23】

レジスタの書き換え要求があった場合の処理手順を示すフローチャート。

【図 2 4】

外部レジスタセット記憶部からレジスタセット群への転送要求があった場合の処理手順を示すフローチャート。

【図 2 5】

レジスタセット群から外部レジスタセット記憶部への転送要求があった場合の処理手順を示すフローチャート。

【図 2 6】

外部レジスタセット記憶部 3 2 のデータ構造を示す図。

【図 2 7】

タスク起動時の処理手順を示すフローチャート。

【図 2 8】

レジスタセット転送部から外部レジスタセット記憶部へのレジスタ転送要求があった場合の処理手順を示すフローチャート。

【図 2 9】

外部レジスタセット記憶部からレジスタセット群への転送要求があった場合の処理手順を示すフローチャート。

【図 3 0】

第 1 1 の実施形態の外部レジスタセット記憶部 3 2 のデータ構造を示す図。

【図 3 1】

外部レジスタセット記憶部からレジスタセット群への転送要求があった場合の処理手順を示すフローチャート。

【図 3 2】

レジスタセット転送部から外部レジスタセット記憶部へのレジスタ転送要求があった場合の処理手順を示すフローチャート。

【図 3 3】

タスク終了処理を示すフローチャート。

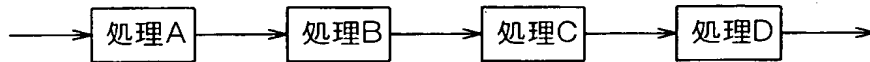
【符号の説明】

1 1 タスク起動部

- 12 実行タスク決定部
- 13 プロセッサコア
- 14 タスクID・優先順位供給部
- 15 実行可能タスクIDリスト
- 16 プライオリティエンコーダ
- 17 時刻計測部
- 18 起動時刻テーブル
- 19 平均時間感覚計算部
- 20 優先順位決定部
- 21 レジスタセット群
- 22 レジスタセット選択部
- 23 演算部
- 24 レジスタセットIDタスクID対応部
- 30 優先順位変更部
- 31 ヒット判定部
- 32 外部レジスタセット記憶部
- 33 外部記憶制御部
- 34 レジスタセット転送部
- 35 退避レジスタセット決定部

【書類名】 図面

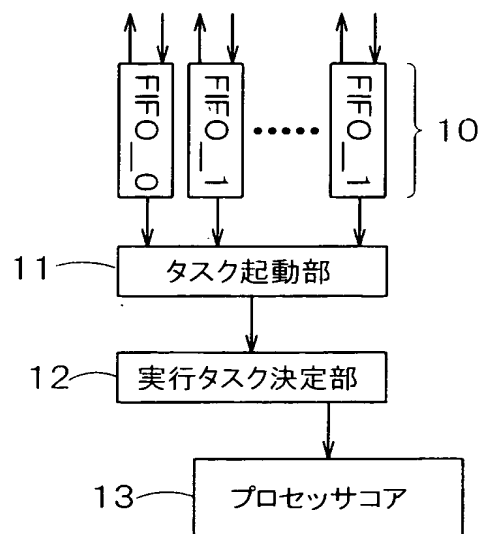
【図 1】



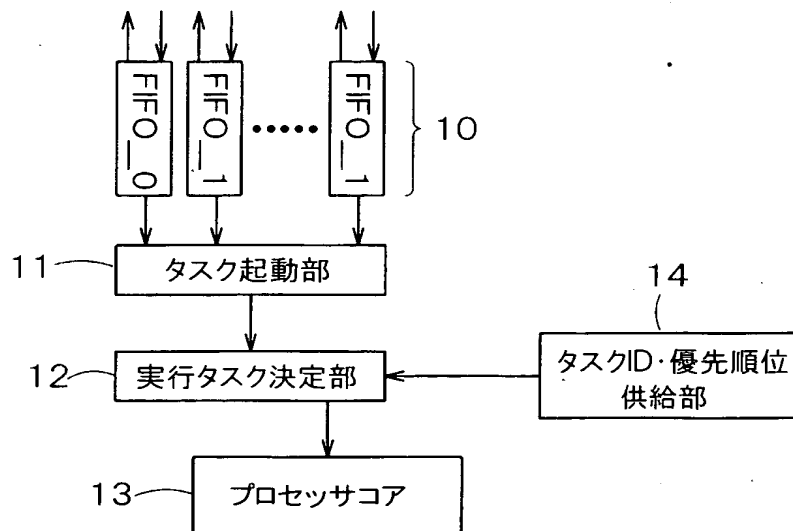
【図 2】



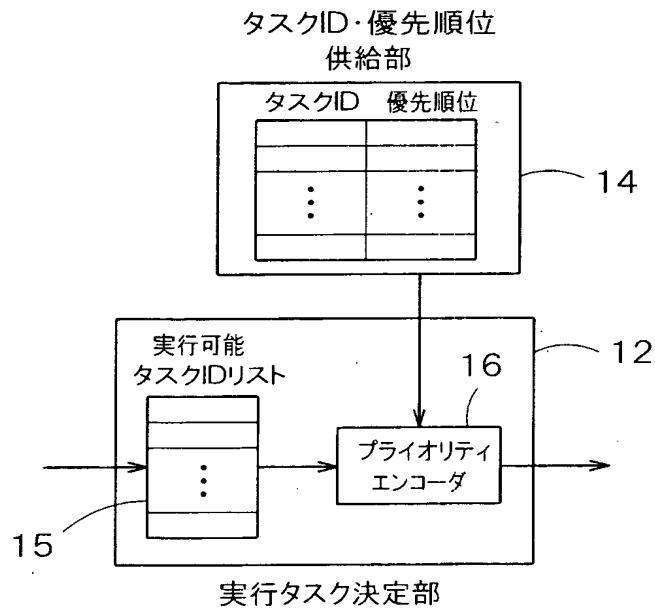
【図 3】



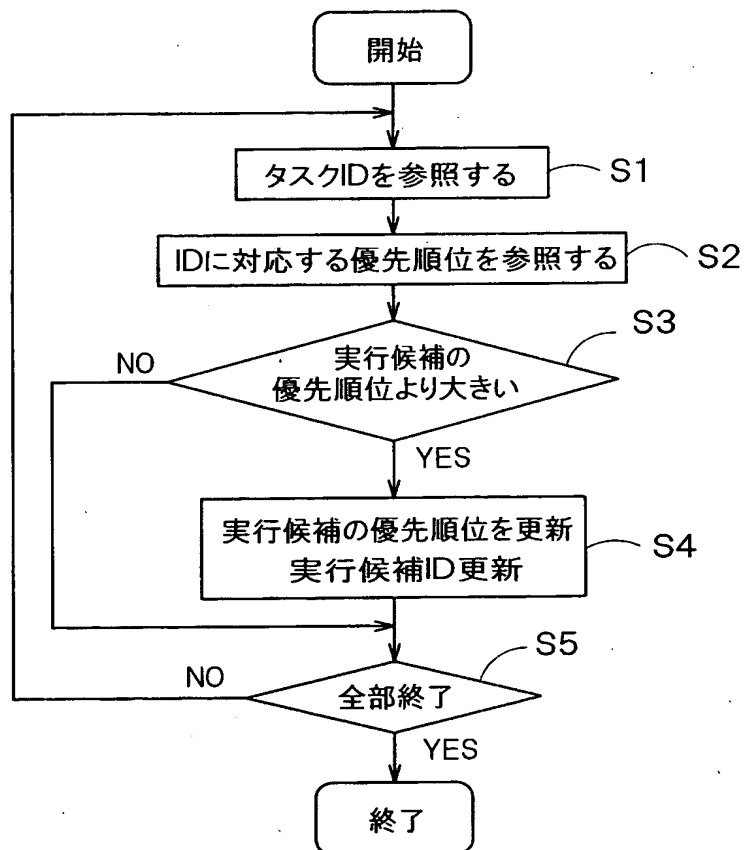
【図 4】



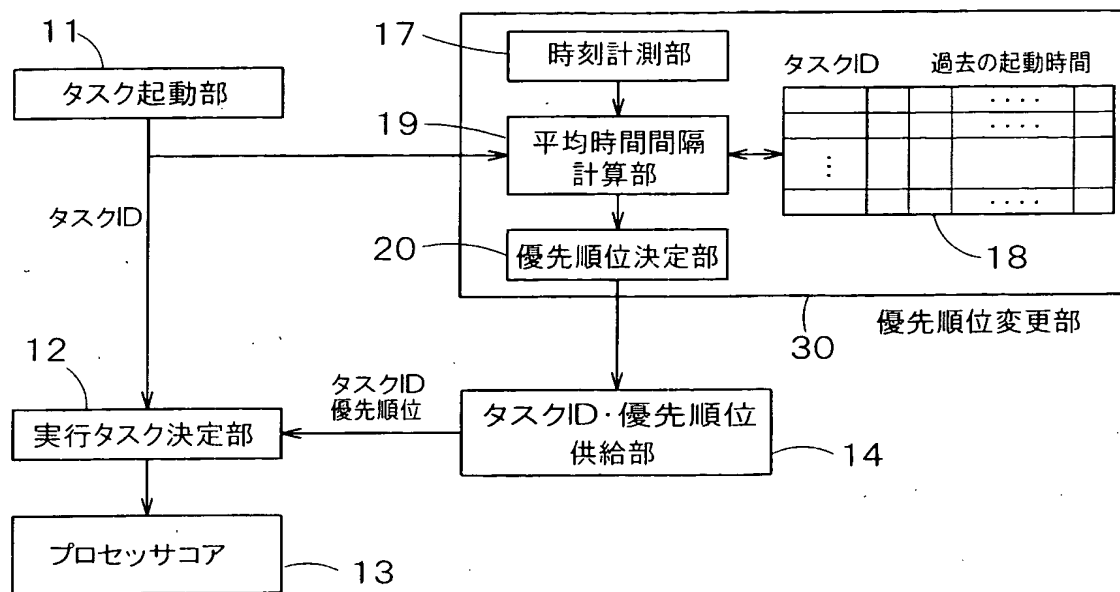
【図 5】



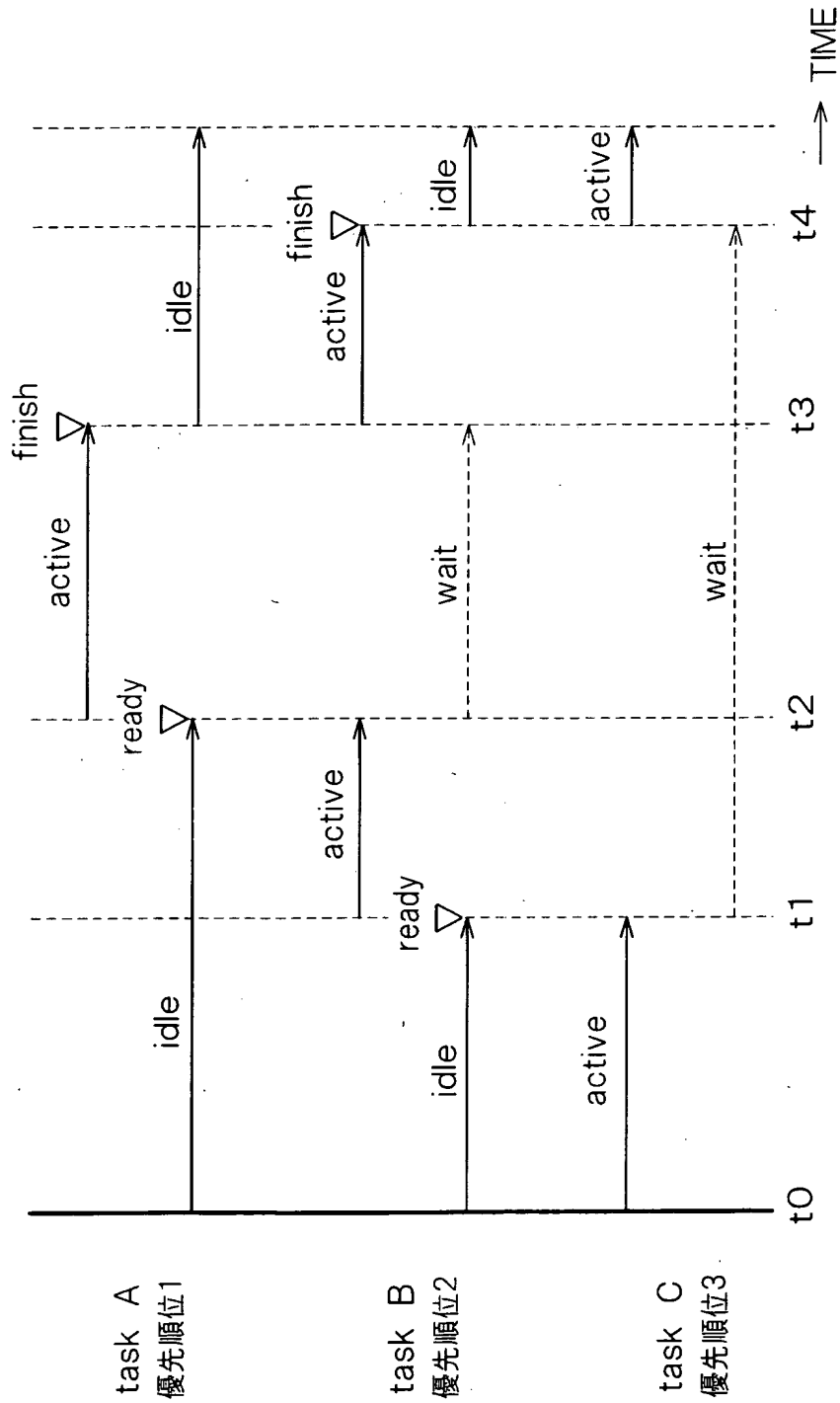
【図 6】



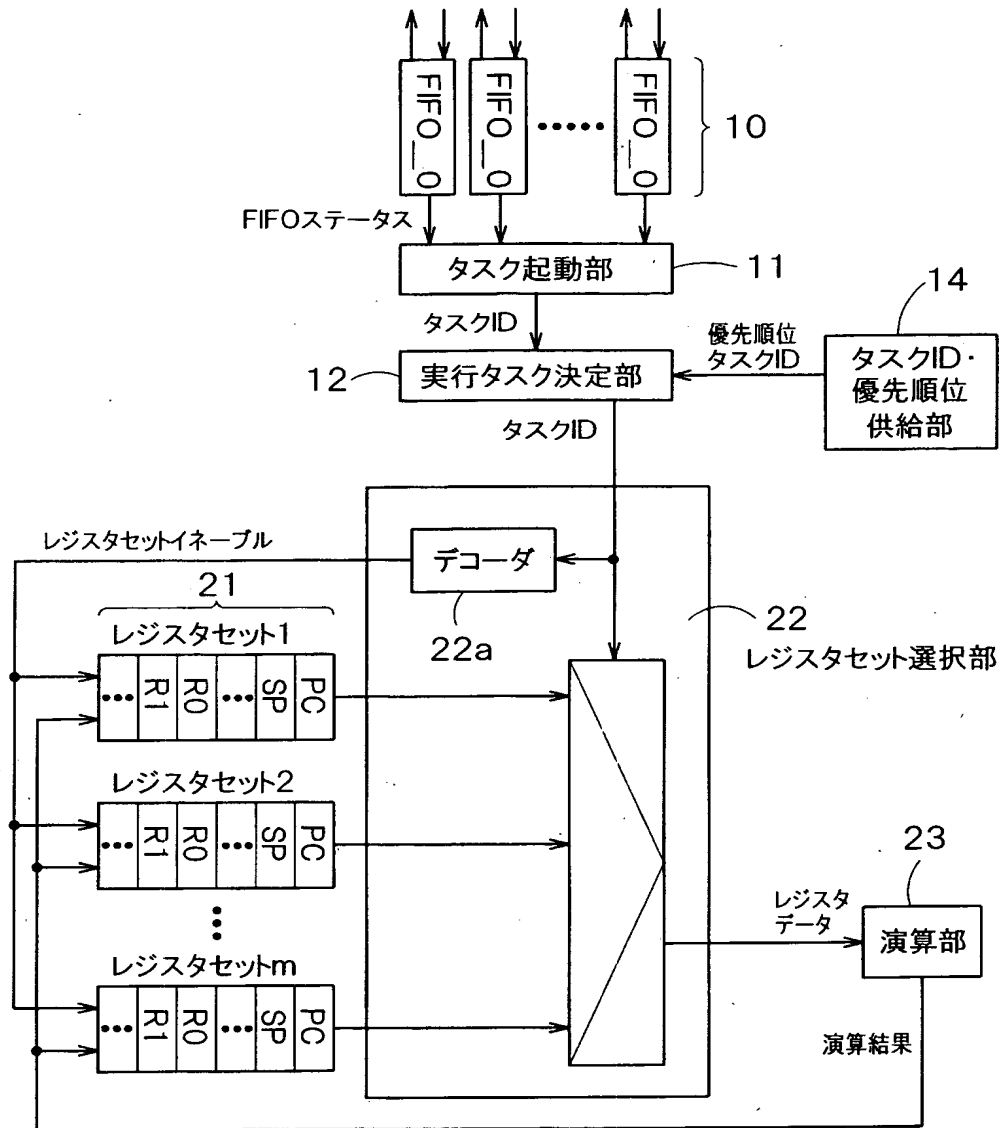
【図 7】



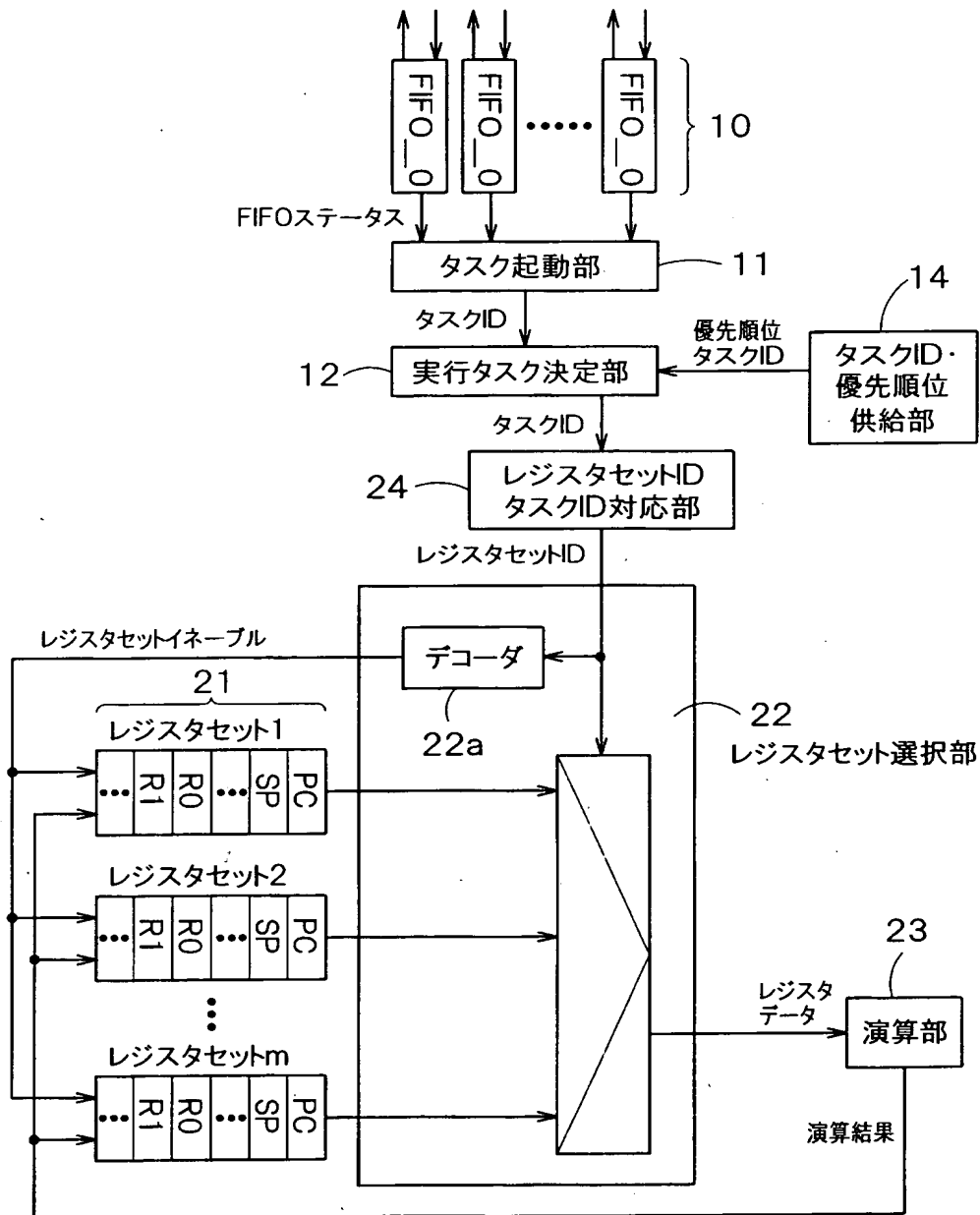
【図 8】



【図 9】



【図 10】

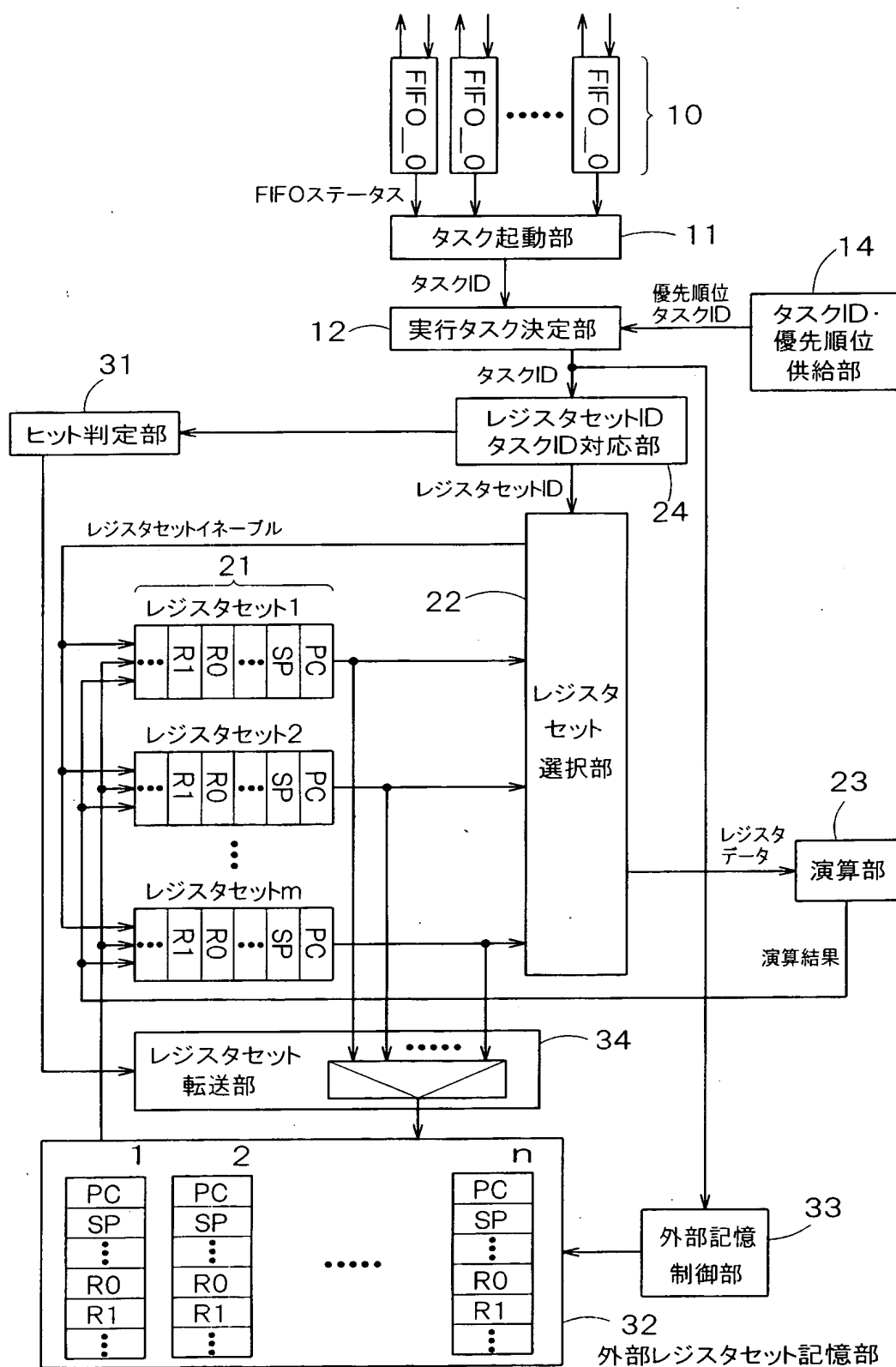


【図 11】

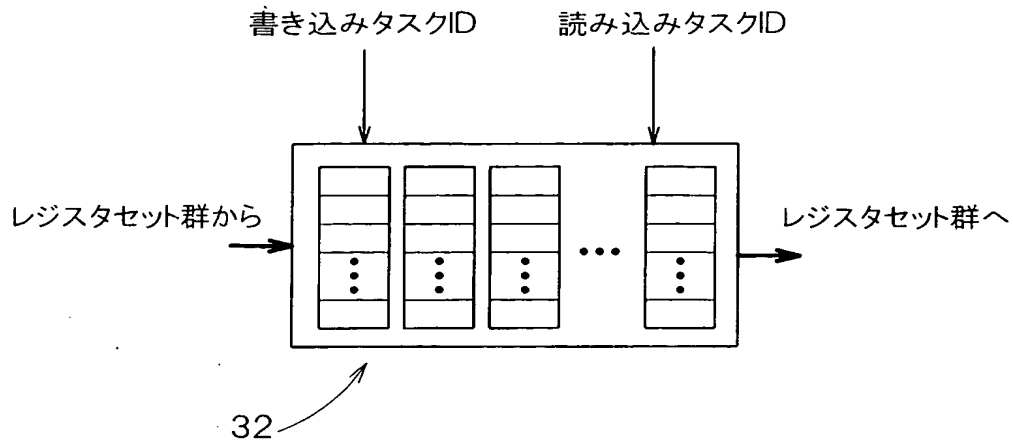
24 →

1	5
2	7
3	10
4	1
⋮	⋮
レジスタセットID → m	← タスクID 2

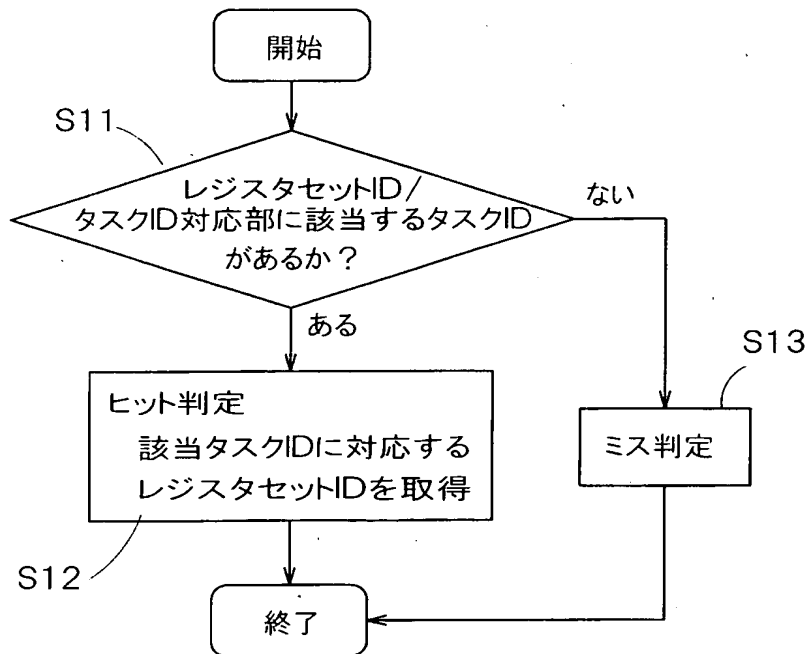
【图 12】



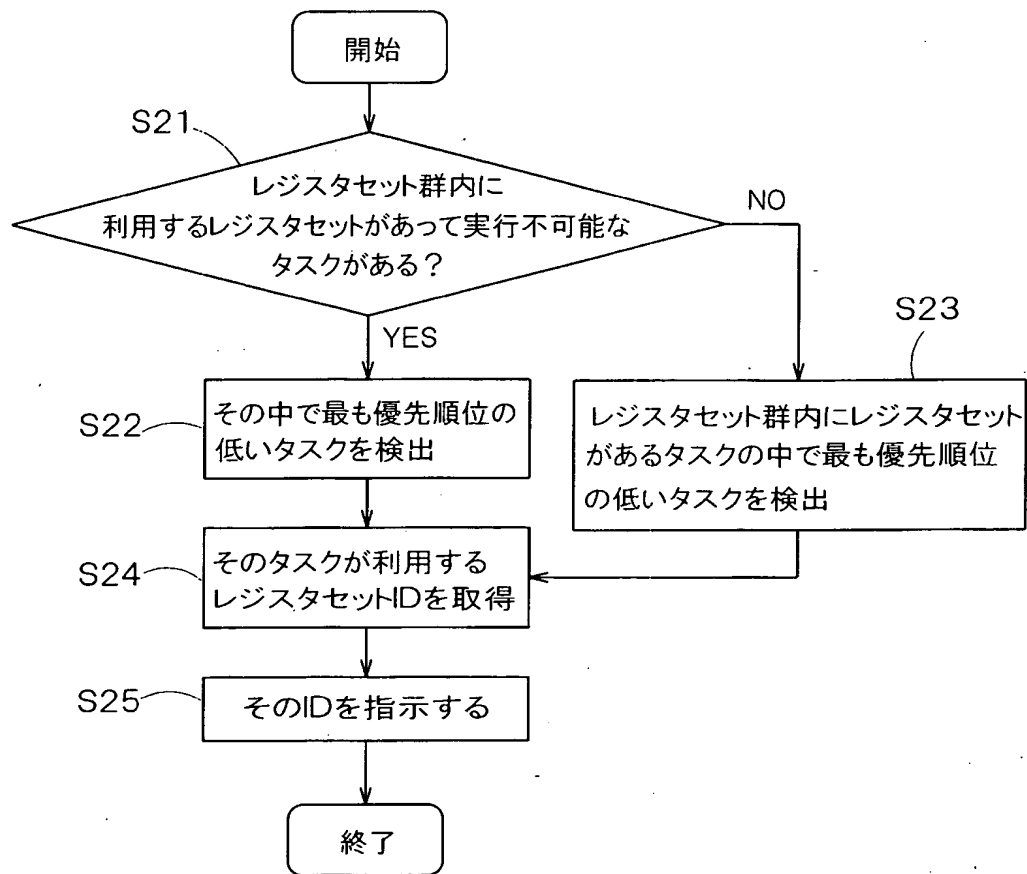
【図 13】



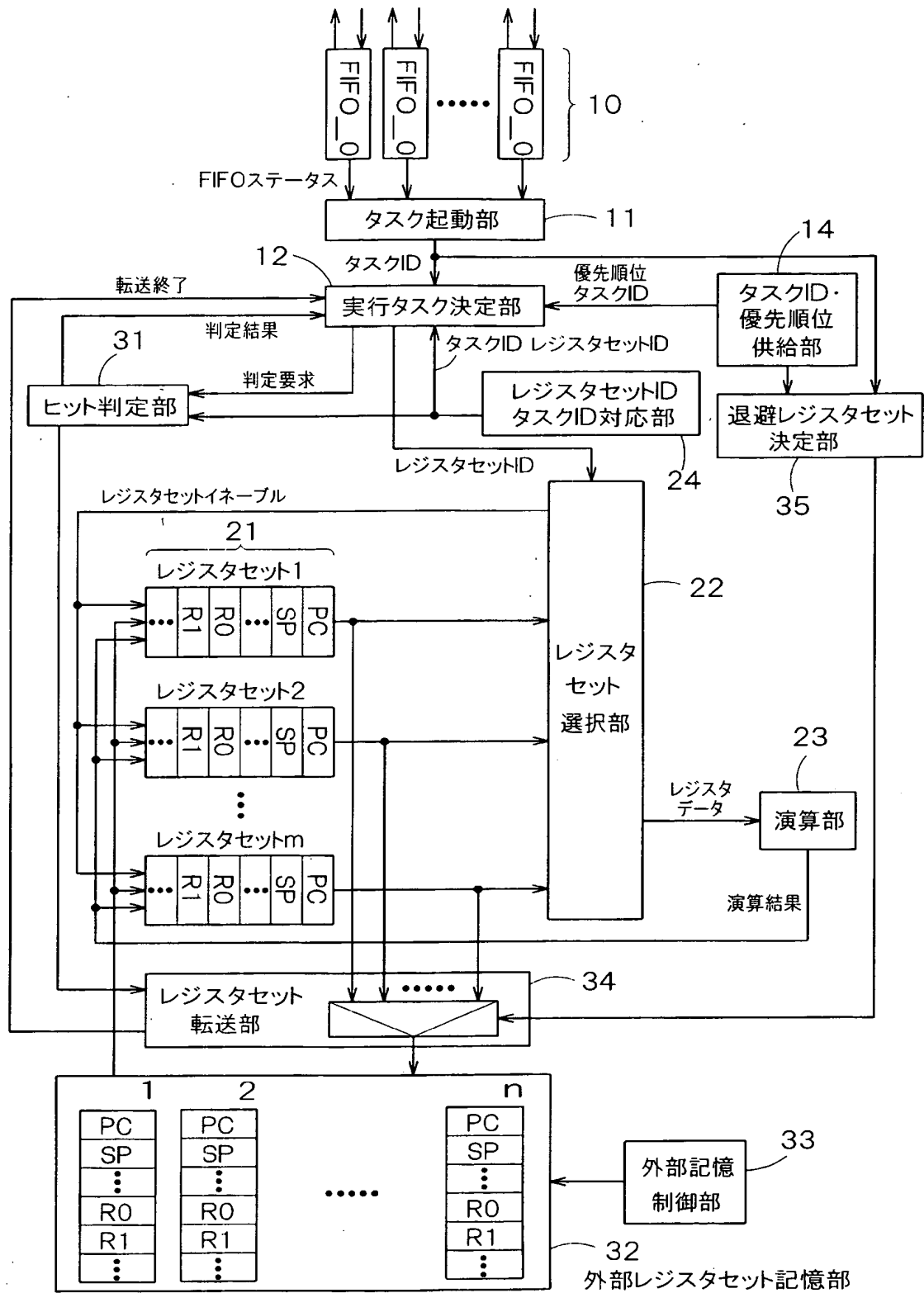
【図 14】



【図 16】



【図 17】



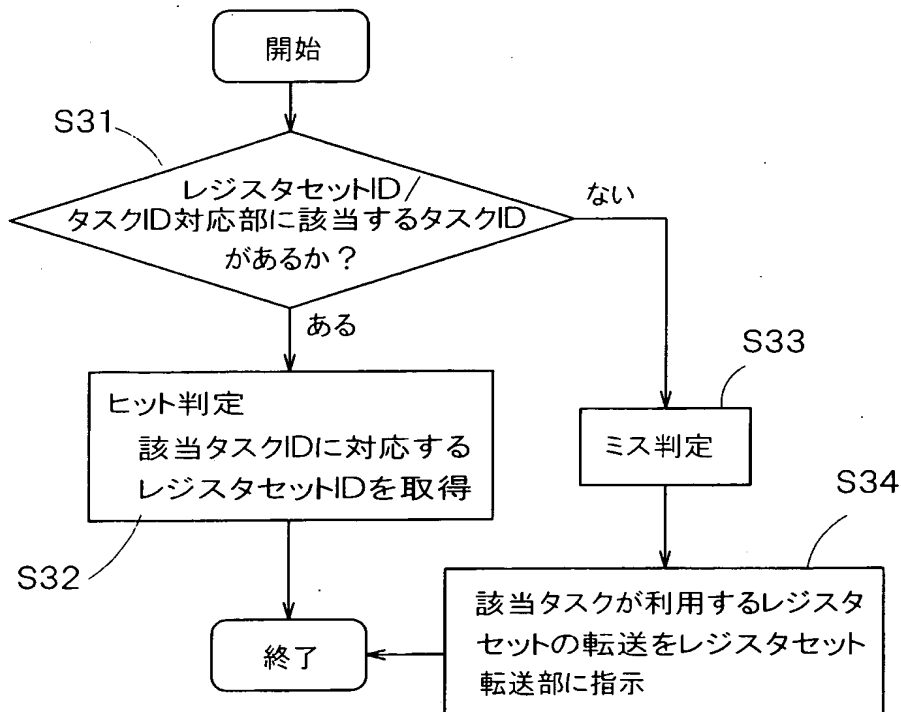
【図18】

24

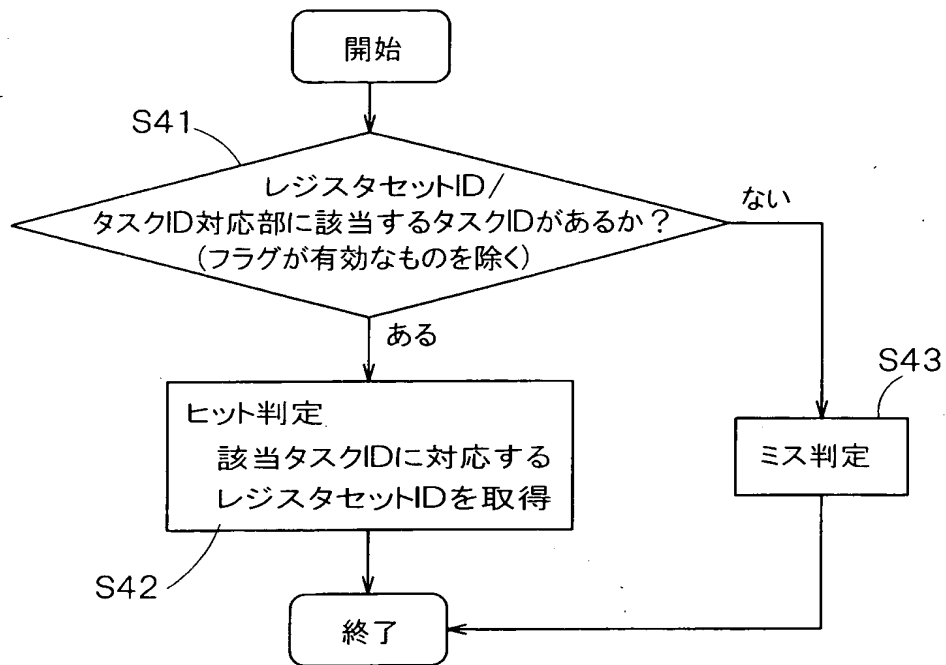
1	5	
2	7	
3	10	✓
4	1	
⋮	⋮	⋮
m	2	

レジスタセットID タスクID フラグ

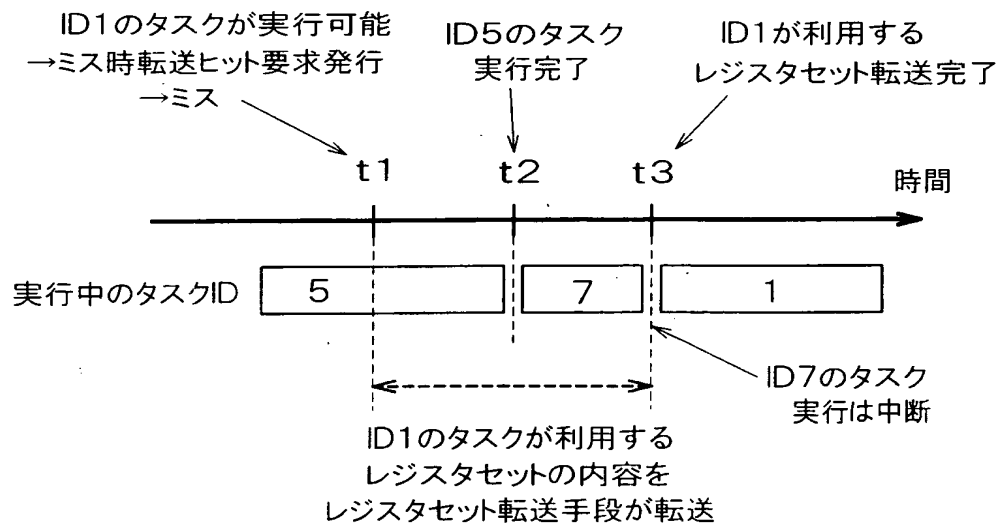
【図19】



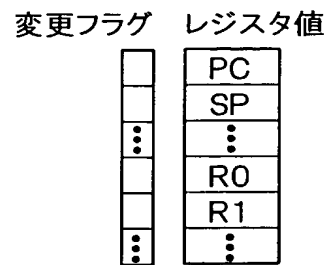
【図 20】



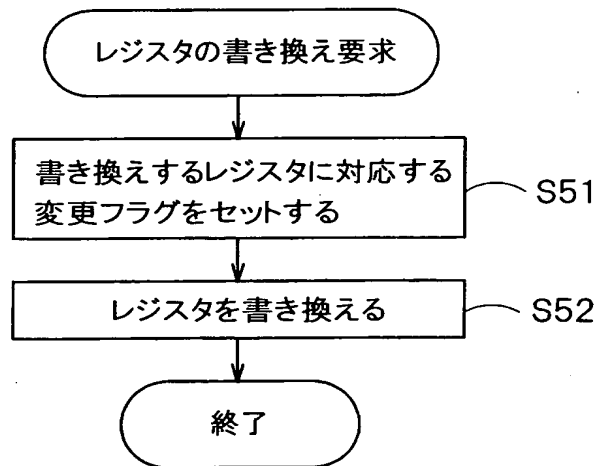
【図 21】



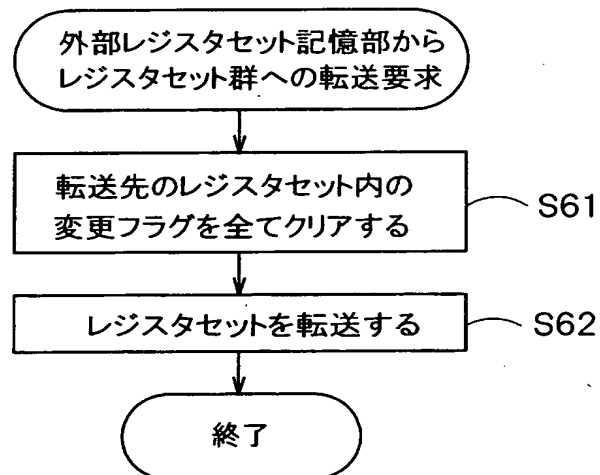
【図 2 2】



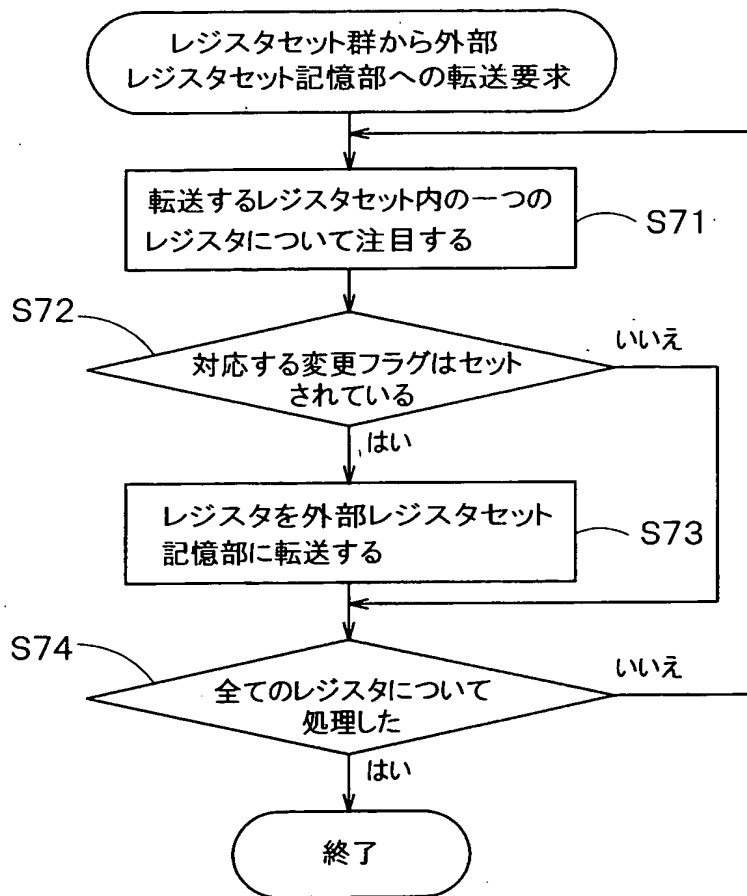
【図 2 3】



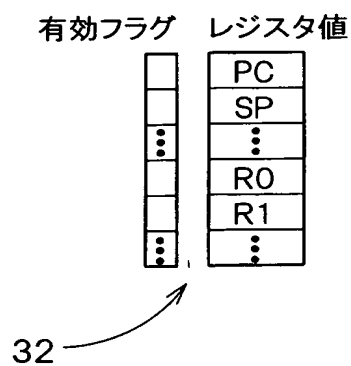
【図 2 4】



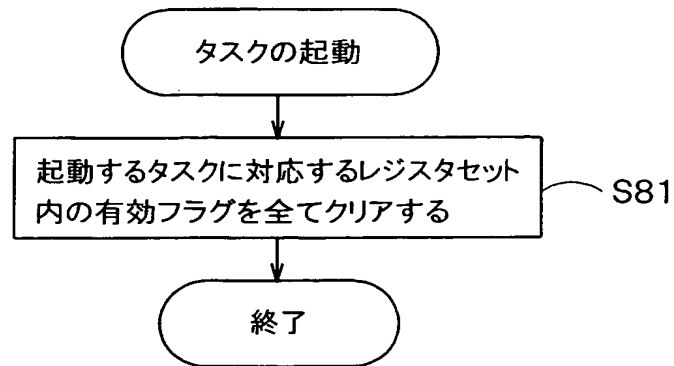
【図 25】



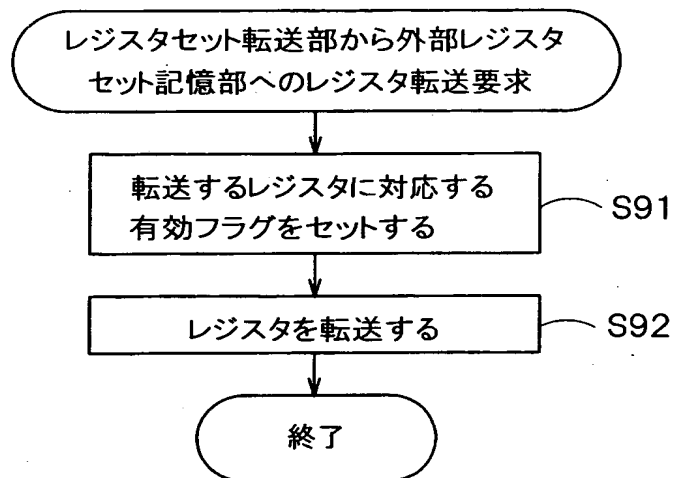
【図 26】



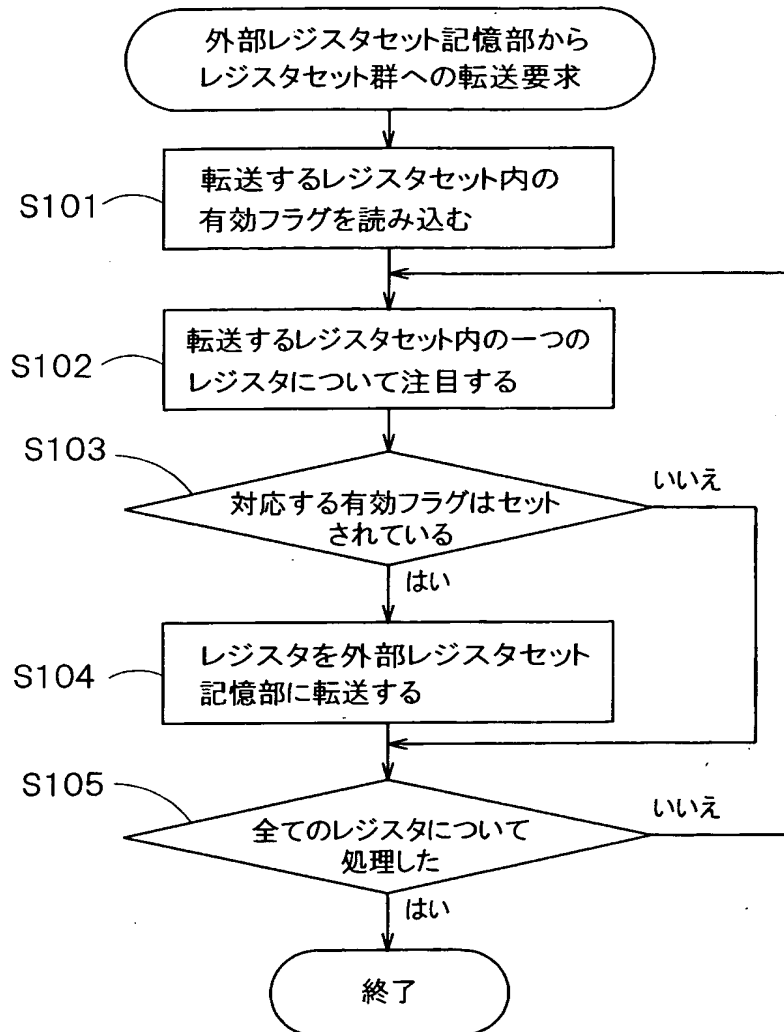
【図 27】



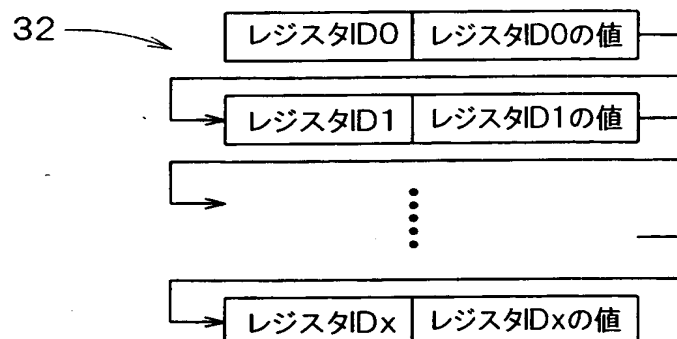
【図 28】



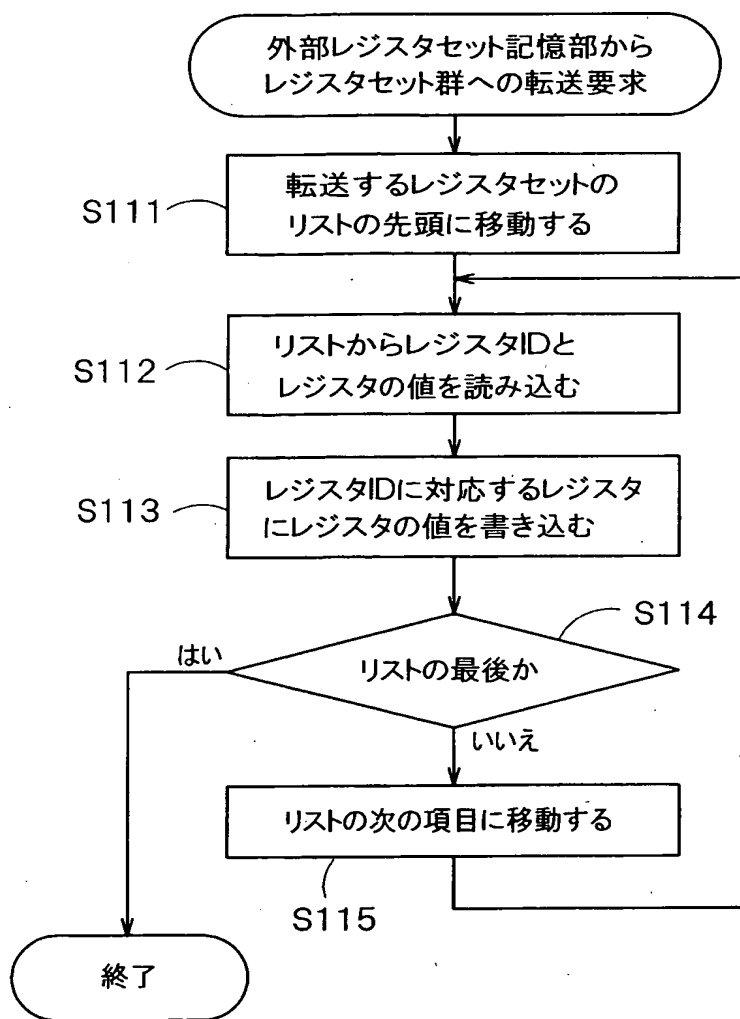
【図 29】



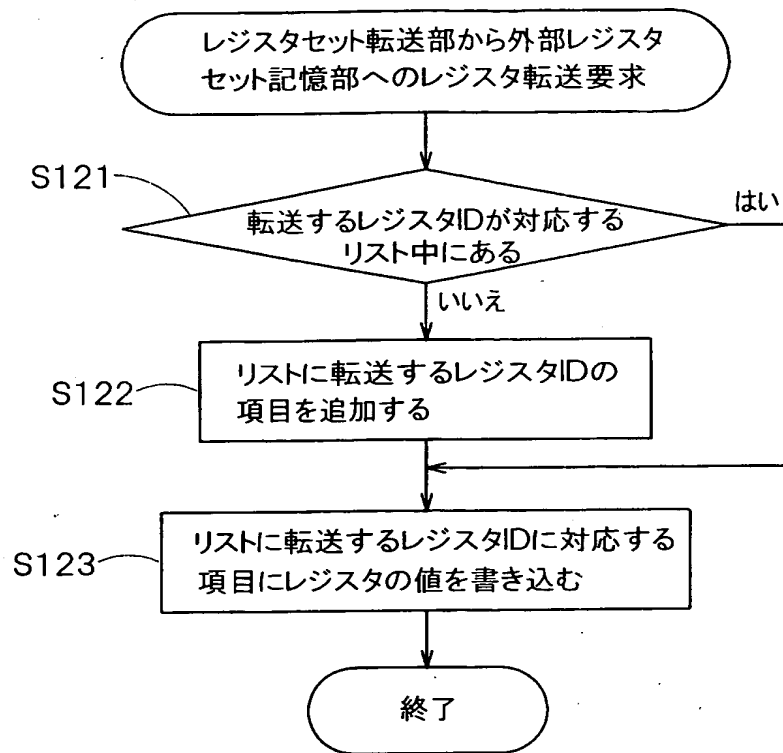
【図 30】



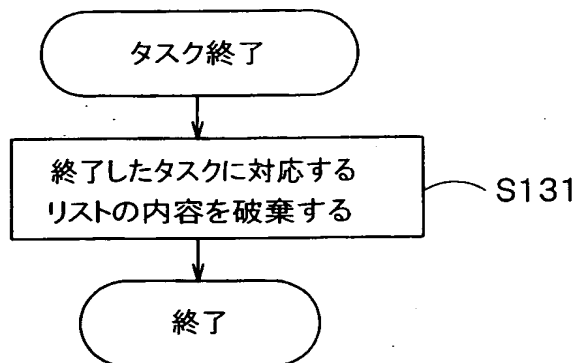
【図 31】



【図 3 2】



【図 3 3】



【書類名】 要約書

【要約】

【課題】 複数のタスクをリアルタイムに効率よく実行できるようにする。

【解決手段】 本発明に係るプロセッサは、複数のタスクそれぞれが起動可能かを判断するタスク起動部 11 と、タスク起動部 11 の判断結果に基づいて起動すべきタスクを決定する実行タスク決定部 12 と、実行タスク決定部 12 で決定されたタスクを実行するプロセッサコア 13 とを備えている。タスク起動部 11 には、複数の FIFO 10 が接続されている。タスクに入力されるべきデータを FIFO 10 が保持し、かつタスクの実行結果を格納すべき他の FIFO 10 に空き領域が存在するか否かをタスク起動部 11 で判断し、その判断結果に基づいて実行タスク決定部 12 が起動すべきタスクを決定するようにしたため、オペレーティングシステムなしでタスクのスケジューリングを行うことができ、オペレーティングシステムのオーバーヘッドがなくなる。

【選択図】 図 3

特願2003-003428

出願人履歴情報

識別番号

[000003078]

1. 変更年月日 2001年 7月 2日
[変更理由] 住所変更
住 所 東京都港区芝浦一丁目1番1号
氏 名 株式会社東芝
2. 変更年月日 2003年 5月 9日
[変更理由] 名称変更
住所変更
住 所 東京都港区芝浦一丁目1番1号
氏 名 株式会社東芝